RSIC-423

SOME MATHEMATICAL PROBLEMS ARISING IN INFORMATION RETRIEVAL FROM INVERTED FILES

by

Robert L. Causey

June 1965

Redstone Scientific Information Center

U S ARMY MISSILE COMMAND
REDSTONE ARSENAL, ALABAMA

Contract No. DA-01-021-AMC-11870(Z)

University of Alabama Research Institute
Huntsville, Alabama

DDC AVAILABILITY NOTICE

Qualified requesters may obtain copies of this report from DDC.

DISPOSITION INSTRUCTIONS

Destroy this report when it is no longer needed. Do not return it to the originator.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

30 June 1965 RSIC-423

SOME MATHEMATICAL PROBLEMS ARISING IN INFORMATION RETRIEVAL FROM INVERTED FILES

By

Robert L. Causey
University of Alabama Research Institute
Huntsville, Alabama

Contract No. DA-01-021-AMC-11870(Z)

EDIS Task II

Director of Army Technical Information
Office of Chief Research & Development
Department of the Army

Information Programs Branch
Redstone Scientific Information Center
Directorate of Research & Development
U. S. Army Missile Command
Redstone Arsenal, Alabama

ABSTRACT

This report is concerned with certain logical and mathematical problems arising in the design and implementation of computer systems for information retrieval from inverted files. The central problem is that of minimizing computer time necessary for mechanized retrieval. Several specific problems are formulated and investigated. Partial or complete answers to some questions are given; but several other questions remain unanswered.

FOREWORD

This study was supported by Redstone Scientific Information Center, U. S. Army Missile Command, under Contract No. DA-01-021-AMC-11870(Z). Mathematical problems associated with retrieval of bibliographic items from an inverted file on digital computer tape were formulated and investigated. The study was generalized so that the results would apply to several different computers.

The University of Alabama Research Institute at Huntsville, Alabama, performed the investigation with Dr. R. L. Causey as the principal investigator, Mr. Claude Martin as the technical monitor, and Mr. Donald Putman as the contract administrator.

The author acknowledges receipt of certain technical information from Mr. W. J. Wilson, General Electric Company, during the course of the investigation.

CONTENTS

Section I. INTRODUCTION

This report is the result of a study concerning possible applications of mathematical reasoning to the improvement of computer retrieval systems for library use.

The principle objectives of the study were to:

1) Investigate in a generalized manner certain aspects of computer retrieval systems which affect the total elapsed time between a request for information and receipt of an answer from the computer.

2) Prove that certain programming techniques involve minimum computer time at least with respect to other techniques which might have been used.

3) Perform such studies without assuming computer characteristics peculiar to a given machine; that is, assume computer characteristics (type of memory, size of memory, logical organization, etc.) which are variable or which are typical and common to most of the medium to large scale digital computers available today. The data to be retrieved were not considered variables. Methods for obtaining a defined bibliographic listing and file organization were subject to choice.

The investigation involved approximately a one man-month effort by the author. The research essentially amounted to a study of the feasibility of achieving objectives such as those outlined in the previous paragraph. At the outset of the study, the author was almost completely unfamiliar with the information storage and retrieval field. This necessitated an initial period of familiarization with basic facts and terminology, and this in turn had an inevitable effect on the final outcome.

Section II. ASSUMPTIONS, DEFINITIONS, AND IDENTIFICATION OF PROBLEMS

It is assumed that the reader is familiar with the general operational characteristics of general purpose digital computer systems of the type commonly being used today, and that he possesses a certain knowledge of the information storage and retrieval field,[1 2] especially that related to library and document analysis systems.

It is further assumed that any retrieval system to be analyzed will be implemented on a general purpose digital computer with the usual logical and arithmetic features, high-speed memory (core memory), several tape handlers for storage of large files on magnetic tapes, and possibly additional intermediate speed random access storage (magnetic drums or disks). No assumptions are made regarding size of high-speed memory (within certain reasonable limits), type of input-output devices or input-output speeds, and buffers associated with tape handlers.

In spite of the wealth of references on library retrieval systems,[3] the author is unaware of precise definitions in print of some of the common terminology used in this field. Notwithstanding the assumption of general familiarity with retrieval systems, therefore, the author yields to a temptation not uncommon among mathematicians to proceed somewhat formally with a few definitions. The definitions should be regarded as tentative and subject to improvements and alterations.

1) Descriptor - A term used by abstractors or library analysts to indicate the contents or subject matter of a document.

2) Item - Any single piece of data (a document number, a descriptor number, an English word, a date, etc.).

3) Record - A group of data items which are all related in some way (all descriptors associated with a given document, all documents associated with a certain descriptor, vital statistics for a single document such as author(s), title, date, source, availability, price and document number).

4) File - A collection of various items and records arranged according to certain rules.

5) Indexed file - A file of records in which each record is identified by an index-heading item.

6) Linear file - An indexed file in which the index is one of a type commonly used for human information retrieval processes (or an indexed file where the index is considered to be of the usual or normal type).

7) Inverted file - An indexed file in which the index is composed of nonindicia items in a corresponding linear file. (For example, a file indexed according to descriptor numbers in which each record is composed of one descriptor number followed by one or more document numbers is the inverse of a file (called linear) in which each record is composed of one document number followed by one or more descriptor numbers.)

8) Request - A logically formulated query for retrieval of information (usually involves only descriptors and certain logical symbols -- see below for examples of such requests).

9) Master file - In an information retrieval system, any file containing all data necessary for retrieval implementation of an arbitrary request.

10) Descriptor frequency - The number of times a given descriptor occurs in a linear master file or the number of documents associated with a given descriptor in an inverted file (c.f. example in No. 7).

11) Tape file - A file stored on a computer magnetic tape.

The main problem to be considered may be described as follows:

Let File M denote a master inverted file in which the records contain one descriptor followed by one or more documents. Let the number of descriptors be much smaller than the number of distinct document numbers (such as in the NASA search system files where approximately 15,000 descriptors are used and more than 100,000 documents analyzed). Let File MB denote a master file (entirely separate from File M) containing all bibliographic data needed by human requestors for every document referenced in File M. Let File M and File MB be stored on magnetic tapes suitable for a given digital computer.

Now let $R_1, R_2, \ldots, R_k$ be a series of requests for all documents in File MB satisfying logical conditions of the following type:

$$\text{(a)} \quad R_m\text{:} \quad A_1 L_1 A_2 L_2 \ldots A_{r_m-1} L_{r_m-1} A_{r_m} \qquad (m = 1, \ldots, k)$$

where the L_i are any of the logical operations

, = logical "and"

+ = logical "or"

− = logical "not"

and the A_i are either descriptors, other expressions of the form (a), or expressions of the form

(b) $(D_1 + D_2 + \ldots + D_n)m$

called majority logic expressions where m and n are positive integers ($m < n$). Any expression of the form (b) has the meaning: the logical "or" of any m of the n descriptors $D_1, \ldots, D_n$. Now we can state the main problem in the form of a question: How does one service the series of Requests $R_1, \ldots, R_k$ on the given computer in the least time?

Auxiliary questions which may be considered are the following: How does the organization of File M and File MB affect the running time? Should we replace File M by its inverse, namely, a linear file? Can we automatically reformulate the requests R_i under certain conditions to give the human originators more useful information?

Section III. SOME RESULTS

It will be assumed that all of the requests $R_1, \ldots, R_k$ can be handled as a group by the computer. The servicing of the requests can be subdivided into four main steps.

1) Consider $R_1, \ldots, R_k$ as an indexed file. Invert it so as to obtain a file indexed by descriptors and a list of all descriptors occurring in the series of k requests.

2) Read File M until all descriptors in the above list have been found; during this read operation store all needed descriptor records in appropriate storage (high speed, intermediate speed, or tape storage in that order of preference). Call the collection of these descriptor records File A.

3) Calculate the document lists satisfying R_i $(i = 1, 2, \ldots, k)$ using File A.

4) Extract the bibliographic information corresponding to the document lists obtained in Step 3 from File MB and communicate same to the appropriate output hardware (printer, visual display unit, remote station, or other device).

We shall first deal with some of the auxiliary questions mentioned in the last section, and follow with a discussion of the main problem.

1) Organization of File M. - This obviously affects only Step 2. Let $D_2 \ldots, D_m$ be the descriptors occurring in $R_1, \ldots, R_k$. Then obviously a best file organization for File M is any one in which $D_1, \ldots, D_m$ (not necessarily in that order) are the first m descriptors occurring in the file. Since the collection $D_1, \ldots, D_m$ is generally different for each series of requests, there is no one best file organization of File M for all occasions. However, since the computer can stop reading File M as soon as all the descriptors $D_1, \ldots, D_m$ have been found, it is evident that those descriptors most frequently used (or those most likely to be used) should come first in File M in order to minimize computer time. In the absence of frequency-of-use (in requests) data, it would perhaps be best to arrange descriptors in File M in descending order of descriptor frequencies.

2) Replace File M by a linear file.- Such replacement would not permit the arrangements discussed in the previous paragraph. One could try to arrange a linear File M in descending order of most frequently used documents, but this seems quite impractical if not impossible. While linear files are usually somewhat shorter than their inverses,[3] it seems that one would have to agree with Costello[3] that the use of inverted files is much more efficient for the type of retrieval operations considered here.

3) Organization of File MB. - It would seem that the records in File MB should be arranged in numerical order of document numbers as a matter of programming convenience. On some machines this might allow skipping over large sections of File MB during execution of Step 4 at tape speeds faster than would be required if the whole file had to be read. The author, however, does not know of any machine permitting forward tape winding at speeds faster than read speeds. On the other hand, if the computer is such that the tape or tapes comprising File MB have to be read until all bibliographic information is obtained, then it might be feasible to arrange the records in File MB in frequency-of-use order. The author does not know how such orderings could be determined in practice.

Step 1 in the servicing process involves essentially only a sorting procedure. The author cannot conceive of a retrieval system in which this step could not be handled entirely within the core storage of the computer. The optimization of this step would involve merely the choice by an expert programmer of the fastest sorting procedure for the particular computer being used.

Of all the steps in the servicing process, Step 3 is the most complex, the most amenable to mathematical analysis, and perhaps the most critical insofar as time-minimization is concerned. The main mathematical tools needed in the study of Step 3 are set theory[4] and Boolean Algebra.[5]

Limitations imposed by the computer's memory capacity (and also memory type, if no random access store is provided) can have a great effect on the running time and on the programming techniques used. Thus if File A has to be stored on a tape, one must resort in most cases to a certain amount of tape-reeling which will be time-consuming. Of course for a computer with buffered tape units, there will always be situations when the necessity of resorting to tape storage will not noticeably slow up the process. It seems likely that tape storage for File A will place limitations on possible applications of Boolean algebra to time-minimization.

Ideally one would like to have a high-speed memory large enough to store File A, the program for executing Step 3, and all intermediate results at any given time during that execution. One would then have maximum flexibility and speed of execution. The next best computer configuration would be one in which all or a part of File A could be stored in a random access memory, which would probably be almost as good as the ideal situation.

Let us turn now to some of the details of Step 3; namely, the evaluation of expressions of the form (a) or, what is the same thing, the determination of all documents in the master file which satisfy certain conditions. We shall assume without loss of generality that each of the elements A_i, if it is not a single descriptor nor a majority logic expression, involves only the logical operation $\cdot$ or the logical operation $+$.

Each descriptor has associated with it a list of document numbers which (for reasons which will be obvious later) are assumed to be arranged in numerical order with the smallest number first. Thus there is a set of document numbers which can be identified with each descriptor.

Consider the subproblem of finding all document numbers satisfying $A = D_1 . D_2 . \cdots . D_n$. Here the D's are descriptors which are arranged in ascending order according to descriptor frequency; that is, the frequency of D_i is less than or equal to the frequency of D_{i+1} for $i = 1, 2, \ldots, n - 1$. Mathematically this problem is equivalent to finding the intersection of the sets of document numbers associated with D_1 through D_n, thereby finding the set of numbers common to each of the n sets. Evidently one can minimize the computer time in determining the intersection by minimizin the number of comparisons of numbers which the computer executes, and can easily obtain a complete solution in the case $n = 2$. Let $N_1, \ldots, N_k$ denote the numbers associated with D_1 and $M_1, \ldots, M_p$ those associated with D_2. We may identify these sets with their associated descriptors, and consider the following six cases.

1) $N_k < M_1$; the intersection is empty and no further comparisons are necessary.

2) $M_p < N_1$; same as Case 1.

3) $N_1 \leq M_1 \leq N_k$, $N_k \leq M_p$; compare only those numbers in D_1 which are $\geq M_1$ with those numbers in D_2 which are $\leq N_k$.

4) $N_1 \leq M_1 \leq N_k$, $N_k > M_p$; compare only those numbers in D_1 which are $\geq M_1$ and $\leq M_p$ with those in D_2.

5) $M_1 < N_1 \leq M_p$, $M_p \leq N_k$; same as Case 3 with the roles of D_1 and D_2 reversed.

6) $M_1 < N_1 \leq M_p$, $M_p > N_k$; same as Case 4 with the roles of D_1 and D_2 reversed.

The procedure outlined minimizes the total number of comparisons and allows for every contingency. The implementation of Cases 3 through 6 with a digital computer program may inevitably involve more comparisons than the required minimum. In fact, on some computers Cases 3 through 6 may involve essentially only one case. One obtains an element of the intersection only when one achieves equality in a comparison. Thus the intersection of D_1 and D_2 may be empty in any of the last four cases; if $n \geq 3$, the situation is more complicated. One could initially carry out (until perchance an empty intersection was discovered) some or all of the $(n - 1)!$ pairs of comparisons such as in Cases 1 and 2 above. This hardly seems worthwhile since most "and" expressions occurring in practice will result in a nonempty intersection. While there is no guarantee of minimizing comparisons, it seems best to carry out the comparisons by looking at only two sets at a time. Starting with those sets having the fewest members seems to be the best procedure. Thus we evaluate A by associating its "factors" as $(\cdot\cdot ((D_1 \,.\, D_2) \,.\, D_2) \,.\, \cdots \,.\, D_n)$.

Now let $A = D_1 + D_2 + \ldots + D_n$ where again the D's are arranged in ascending order of frequency. Here the problem is one of determining the set theoretical union of n sets. Consider first the Case $n = 2$. The union is composed of all numbers in the two sets with the duplications (i.e. the numbers in the intersection of the two sets) thrown out. Thus the problem of constructing the union is closely related to that of determining the intersection. The author believes that a single subroutine could accept two sets as input and deliver either the union or intersection as output at the user's option. Essentially the same comparisons are involved in either case; the only difference is in the selection of numbers for output. In the case of $n \geq 3$, one probably should proceed as in the evaluation of intersections with some assurance of saving work because of the ascending order of frequencies.

The problem of evaluating an expression of the form $A_1 - A_2$ is that of writing down all numbers in the set A_1 which are not in A_2. Removing from A_1 those numbers in the intersection of A_1 and A_2 is all that is needed. Again the same subroutine which computes either an intersection or a union could be generalized to output the "difference" of two sets. A very slight gain in speed could be obtained by having three separate subroutines for intersection, union, and difference, but the saving does not seem worthwhile.

The evaluation of majority logic expressions is somewhat more complex because the output is $C(n, m) = n!/m!(n - m)!$ sets instead of a single set. The symbol $C(n, m)$ is the familiar one denoting the number of combinations of n things taken m at a time. The presence of a majority logic expression thus implies that the request in which it occurs is really several requests in one. It would appear that majority logic expressions involving large values of n should be avoided whenever possible.

The auxiliary questions regarding reformulation of requests is related to majority logic expressions. Suppose a request, R, results in zero or only a few documents. Two ways of modifying the logical expression for R which will increase the number of documents in the final output are to omit a descriptor in a logical "and" expression, and to reduce the size of the second set, B, in a difference expression $A - B$. If B is a logical "or" expression, either omitting descriptors from it or, what is more general, converting it to a majority logic expression will accomplish this end. The author will not attempt to suggest general rules which should be applied in such situations, but the matter appears to merit investigation.

Section IV. RECOMMENDATIONS FOR FUTURE WORK

The discussion contained in Section 3 is obviously incomplete because many questions are not answered or answered only by conjecture. There are other questions which were not even raised. For example: Can one manipulate the requests by the rules of Boolean algebra before they are processed in order to effectively reduce running time? In what order should one evaluate the expressions A_i in (a)? How does the ratio of available data storage space in core to the size of File A (when it cannot be stored in core) affect running time? (The author conjectures that the ratio has very little effect as long as one can always store at least three of the sets which occur during evaluation in core at the same time.) Can one establish useful rules for modifying requests as suggested in the last paragraph of Section 3? Are there situations in which a linear File M would be preferable from a timesaving or other standpoint?

It is clear that investigations of the sort attempted in this report could be extended and other questions answered. Evidently any research effort in this direction would be a technical one. Many questions could be investigated by a nonmathematician who is knowledgeable in library science and good at logic. On the other hand, manipulations of Boolean expressions and set theoretical statements may require the services of someone with mathematical training. It is the author's feeling that the problems considered in this report are far too involved to permit complete solution in the space of one or two months. The limitations which working time have imposed on the results of this study were inevitable; but it is hoped that the way to more complete results has been illuminated.

LITERATURE CITED

1. Becker, J. and R. M. Hayes, INFORMATION STORAGE AND RETRIEVAL, New York, John Wiley and Sons, Inc., 1963.

2. Costello, J. C., Jr., COMPUTER REQUIREMENTS FOR INVERTED COORDINATE INDEXES, American Documentation, 1962, vol. 13, pp. 414-419.

3. Markuson, Barbara E., Ed., LIBRARIES AND AUTOMATION, Library of Congress, Washington, D. C., 1964.

4. Halmos, P. R., NAIVE SET THEORY, Princeton, Van Nostrand, 1960.

5. Hohn, F. E., APPLIED BOOLEAN ALGEBRA, New York, The Macmillan Company, 1960.

SELECTED BIBLIOGRAPHY

1. U. S. Naval Ordnance Laboratory, White Oak, Maryland, NOL RETRIEVAL SYSTEM FOR THE 7090 by J. J. Crockett, 1 October 1964. Rep. NOLTR 64-125.

2. The MEDLARS STORY at the National Library of Medicine, Public Health Service, U. S. Department of Health, Education and Welfare, 1963.